Handling data — Part 1 — Elements of correction

1 With two arrays

For the first questions, the skeleton code could be modified as in the code given in Listing 1 :

```
1  import unicodedata
2  # This removes diacritics: accents, diaereses, umlauts, tildes, cedillas...
3  def normalize(text):
4      nfkd_form = unicodedata.normalize('NFKD', text)
5      return "".join([c for c in nfkd_form if not unicodedata.combining(c)])
6
7  def normalizeAndLowercase(text):
8      return normalize(text).lower()
9
10 names = []
11 phones = []
12
13 # "iso-8859-1" is a common encoding. On Linux, the standard encoding is
14 # "utf-8" and on Windows you can also encounter "cp1252".
15 f = open("TP9_Contacts.txt", "r", encoding="iso-8859-1")
16 # strip() removes blank characters at the beginning and the end of the string,
17 # here in particular the end of line characters left by readline()
18 while(True):
19     name = f.readline().strip()
20     if (name == ""):
21         break
22     phone = f.readline().strip()
23     if (phone == ""):
24         print("A contact has been found without number.")
25         break
26     names.append(name)
27     phones.append(phone)
28 f.close()
29 nbContacts = len(names)
30
31 s = input("What contact do you want to search for ? ")
32 for i in range(nbContacts):
33     if (normalizeAndLowercase(s) == normalizeAndLowercase(names[i])):
34         foundContact = True
35         print(names[i] + " has number " + phones[i] + ".")
36 if (not(foundContact)):
37     print(s + " is not in your contacts.")
```

Listing 1: Handle our contacts file — Two arrays — http://www.barsamian.am/2023-2024/S6ICTB/TP9_Contacts_bis.py.

First, we make sure to compare normalized and lowercase versions of the name given by the user and given in the contact file. Don't forget to normalize and lowercase both variables. You want to find your contact Bob even if you wrote "Bob" in the file and typed "bob" when prompted, but also if you wrote "bob" in the file and typed "Bob" when prompted!

Then, we make sure that our contact list can be as big as wanted. To do that, instead of creating an array of capacity 10 and filling it from cell 0 to 9, we create empty arrays and make them grow with the "append" method. This avoids having an "out of range" error (that happens when we try to access a cell of an array with a number outside of the range $\{0, 1, 2, \dots len(array) - 1\}$).

Last, we process the array of contacts with a slightly different condition. Instead of processing it until we found a suitable contact, we process it from the beginning to the end. Each time we encounter a suitable contact, we print it. At the end of the processing, how to know if a suitable contact was found? This is the role of the "foundContact" variable.

Another way to do it is to count the number of lines in the file first, see Listing 2. The number of contacts is then simply half the number of lines:

```
...

f = open("TP9_Contacts.txt", "r", encoding="iso-8859-1")
nb_lines = 0
for line in f:
    nb_lines += 1
nbContacts = nb_lines // 2
names = [""]*nbContacts
phones = [""]*nbContacts

...
```

Listing 2: Count the number of lines.

2 With a dictionary

For question 6, the skeleton code could be modified as in the code given in Listing 3 :

```
import unicodedata
# This removes diacritics: accents, diaereses, umlauts, tildes, cedillas...
def normalize(text):
    nfkd_form = unicodedata.normalize('NFKD', text)
    return "".join([c for c in nfkd_form if not unicodedata.combining(c)])

def normalizeAndLowercase(text):
    return normalize(text).lower()

contacts = {}

# "iso-8859-1" is a common encoding. On Linux, the standard encoding is
# "utf-8" and on Windows you can also encounter "cp1252".
f = open("TP9_Contacts.txt", "r", encoding="iso-8859-1")
# strip() removes blank characters at the beginning and the end of the string,
# here in particular the end of line characters left by readline()
while(True):
    name = f.readline().strip()
    if (name == ""):
        break
    phone = f.readline().strip()
    if (phone == ""):
        print("A contact has been found without number.")
        break
    contacts[name] = phone
f.close()

s = input("What contact do you want to search for ? ")
foundContact = False
for name in contacts.keys():
    if (normalizeAndLowercase(s) == normalizeAndLowercase(name)):
        foundContact = True
        print(name + " has number " + contacts[name] + ".")
if (not(foundContact)):
    print(s + " is not in your contacts.")
```

Listing 3: Handle our contacts file — Dictionary — http://www.barsamian.am/2023-2024/S6ICTB/TP9_Contacts_ter.py.

First, we build a dictionary instead of two arrays. We start with an empty dictionary `{}` and we append a new contact by simply updating thanks to `contacts[name] = phone`, as we would do with arrays. The only difference is that here, we access the dictionary directly by the key we need, instead of needing an index.

Then, we see that it is not possible to put twice the same key inside the dictionary. How can we solve this problem?

A first idea would be to put an array of strings in the dictionary, not a string. Unfortunately, a dictionary can only hold strings, not arrays.

A second idea is to add the entries in the dictionary with slightly different names. Instead of adding "Bob" each time, we would add "Bob", then "Bob1", "Bob2", etc. in the dictionary. This poses a challenge: how to track the number of times we added "Bob(something)" in the dictionary?

It is possible to do it with a loop. See the modified code given in Listing 4. Please download this code and test it on different contact files (for example, with multiple "Bob").

```
1  import unicodedata
2  # This removes diacritics: accents, diaereses, umlauts, tildes, cedillas...
3  def normalize(text):
4      nfkd_form = unicodedata.normalize('NFKD', text)
5      return "".join([c for c in nfkd_form if not unicodedata.combining(c)])
6
7  def normalizeAndLowercase(text):
8      return normalize(text).lower()
9
10 contacts = {}
11
12 # "iso-8859-1" is a common encoding. On Linux, the standard encoding is
13 # "utf-8" and on Windows you can also encounter "cp1252".
14 f = open("TP9_Contacts.txt", "r", encoding="iso-8859-1")
15 # strip() removes blank characters at the beginning and the end of the string,
16 # here in particular the end of line characters left by readline()
17 while(True):
18     name = f.readline().strip()
19     if (name == ""):
20         break
21     phone = f.readline().strip()
22     if (phone == ""):
23         print("A contact has been found without number.")
24         break
25     if name in contacts:
26         i = 1
27         while (name + str(i)) in contacts:
28             i = i + 1
29         contacts[name + str(i)] = phone
30     else:
31         contacts[name] = phone
32 f.close()
33
34 s = input("What contact do you want to search for ? ")
35 foundContact = False
36 for name in contacts.keys():
37     if (normalizeAndLowercase(name).startswith(normalizeAndLowercase(s))):
38         foundContact = True
39         print(name + " has number " + contacts[name] + ".")
40 if (not(foundContact)):
41     print(s + " is not in your contacts.")
```

Listing 4: Handle our contacts file — Dictionary enabling multiple values for a key — http://www.barsamian.am/2020-2021/S6ICT/TP9_Contacts_quater.py.